RISC-V Encryptor – Progress Report

# 02/08/2022

I've gotten the core to run on the DE10-Lite and I ran C program on it. I've used the preexisting AI accelerator to multiply numbers. I'm now researching ways to implement AES. I read about two possible ways to implement AES:

1. Implementing the four stages
2. Using lookup tables as specified in [this paper](https://pdf.sciencedirectassets.com/277348/1-s2.0-S1875389212X00063/1-s2.0-S1875389212005822/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEGcaCXVzLWVhc3QtMSJGMEQCIFc4wQqU9AgyPceOWl5h%2BNXxrjvbNA%2FovU2O3zcEsMBuAiBkHviAJUNO9d8VCf72dk1zLiwSidhE9jvg%2Bzhnf5wU3SrVBAjv%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F8BEAUaDDA1OTAwMzU0Njg2NSIMNCfCICixS1NXDNl3KqkE2Uhy3CC8wUZaNGQgNqcePEZma13PjQC5q2eBnACvdq8N7QZWKbAiqZ4vW1RBstNBr2fP5q4ZXfsXkT3d99QBGTCtPRttRrs2XMzQgoVqwuCa6u398I1lRCulR%2Fh7oG%2Fn3sSzmurj7ffADZ78BMAY3a1bq%2FyKFDjX9%2BwCDU38FxYhpkBvIe486VEGpF6X9HkXxNwPCPxbN5P0NZqMOqPoX%2Fwx32CusHyC%2FqTIc%2B4axtsqWixzP7s0mkzMnH32UPNn8g8HfbfOKFagNFgVS3%2FEg4Tw%2FI5DARKQuj8wJdZW8zv%2B0F%2B5sqxxi2HsIhqlhW%2FDNdoBu4fPDeJggHKDcLwgW2noLgjxqiwpK1ItRNBWVmEWQ2HPjrY3qvuMoqH9PWXB5O9RiIbFeyJ4w2ne%2FMory2Di%2BVjXzIrw%2F5Wxk4Ll72aJ1qU0zSG7PJCc5ETjK2rAxesKVlvZyU5kWITTRmXXlR0AgXegsOqIyan0%2FOC9j9%2FXGnVCKvXGRx8cjxR1YP0w%2BNDio%2By1%2Bl%2BgpL%2F524DLz6ZFNsp3bTNU7%2FwC9t65FNd2nIXx2p62sdeXn0ohdSpE3L5HhhUPAzmaiklDLyqaJC976p02mfQ9MUbNAZ%2BHrmx2yidZ7rQyoj7xkjyYky4%2F3qjKMLpppPeIU95tLMC6hGscJbNHo4o2DuAD0PDWhX7N4vtYDPgZapE0PuIbgGS%2FfNW0HiuqVt8eOkcwASYjjpEkrLcg0uoy3zCJqN2YBjqqATpa2z7TN8enDChypoucPdCD3mD2bBmbrhMGalk8nws0lB2m%2FCYkdFGEVId5e%2BLkjB6XtiB0K%2FsJLHgtTevNGmGQKW9Dep9vgmNA9RLDbH8NewO3xDGQOpz4I3tuvVGyzbu4pkKmXWTxd%2B6j3wIZKRabTjsv%2FIbbRKDKq%2FMtlhcXBRSEBbRcrc4793YFIolkrwVypTR3D0wA53O4nVmG%2FfQk06gHNJYe%2FZG5&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20220906T145632Z&X-Amz-SignedHeaders=host&X-Amz-Expires=300&X-Amz-Credential=ASIAQ3PHCVTY7KJUTUJE%2F20220906%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=bf18e2aa6ef4c4788acdba4ef1252dc735f12b3775f434f762fa2ae21840df4d&hash=a22f313ffe89139680c0f65831238d3df678e1b8cb2541cde13929c14460ed15&host=68042c943591013ac2b2430a89b270f6af2c76d8dfd086a07176afe7c76c2c61&pii=S1875389212005822&tid=spdf-61737145-0b55-4f23-8248-159ef4d35630&sid=5439c3f85e16c44bf76bcac5a4065666cb7fgxrqb&type=client&ua=4d575656560754545b07&rr=74680d475a5594dd)

I’m currently researching the lookup table method, as it sounds faster and easier to implement.

# 06/09/2022

I’ve started implementing AES encryption using lookup tables in python. Some parts of the implementation are based on the paper specification above, and some are based on [this go language library code](https://github.com/golang/go/blob/master/src/crypto/aes/block.go).

Code skeleton was taken from [this GitHub repository](https://github.com/boppreh/aes)

# 07/09/2022

I’ve successfully implemented AES in python using lookup tables. Currently, the code isn’t pretty, but it’s functional.

# 08/09/2022

I’ve made the preexisting AI accelerator bigger and allocated more memory to MMIO. It currently receives 2 128-bit values each as an array of 4 4-byte values. I’m testing a 4-wide vector multiplication unit where each scalar is one byte.

The code isn’t working. When multiplying the vector by the vector the FPGA reports that the multiplication result is 0.

# 11/09/2022

I’ve found multiple bugs when trying to setup a simulation, they are now fixed but the accelerator still isn’t working. I’m working on a simulation to find more bugs.

# 12/09/2022

Using the simulation, I’ve fixed the vector accelerator. See screenshot of the results below  
Graphical user interface, application, table

Description automatically generated